(NeXT Tip #37b) long long int

Christopher Lane (*lane[at]CAMIS.Stanford.EDU*)
Wed, 7 Jul 1993 15:37:10 -0700 (PDT)

If you managed to suffer through the previous half of this tip, you might be left wondering, "Just what did he mean by a 'long long int'?" We all know the basic 'C' integer types and how they are implemented on 32-bit machines:

type bytes maximum unsigned value

unsigned char 1 255 (UCHAR_MAX)
unsigned short int 2 65535 (USHRT_MAX)
unsigned int 4 4294967295 (UINT_MAX)
unsigned long int 4 4294967295 (ULONG_MAX)

(I used unsigned types but it doesn't really matter for this discussion) In some 'C' implementations, there's an additional, longer type:

type bytes maximum unsigned value

unsigned long long 8 18446744073709551615

(Personally, I think this is what 'long' should have been but that's another story.) This 64bit integer can be signed or unsigned, as needed.

Places where this additional type can be useful are in large number arithmetic, hash or encryption keys, bit patterns, etc. However, I don't recommend using it unless you absolutely need it. This is due to potential portability problems and for lack of support for it.

For example, the 'printf' manual page explains the proper way to print an int (%d), a short signed int (%hd), a long unsigned int (%lu), etc. but there is nothing to deal with 'long long' types -- you have to invent your own printing routines. Also, 'gdb' on the NeXT doesn't seem to handle them correctly.

And, as you probably guessed, there is a similar type for floating point:

type bytes maximum positive value

float 4 3.4028234663852886e38 (MAXFLOAT)
double 8 1.7976931348623157e308 (MAXDOUBLE)

long double 8 1.7976931348623157e308

But 'long double' is the same size as 'double' and so isn't too useful. (And probably should have been called 'double double', IMHO.)

- Christopher